PBS Job scheduling for Linux clusters

Presentation overview

- Introduction to using PBS
- Obtaining and installing PBS
- PBS configuration
- Parallel jobs and PBS
- The MAUI scheduler
- The mpiexec parallel job launcher

Users' working model

What we want users to do:

- Prepare programs
- Understand resource requirements of their programs
- Submit a request to run a program (job), specifying resource requirements, to a "referee", a system program which will execute them in the future, according to the site policy, to best satisfy the site production needs

We can control the "referee" program (batch system) => we have (potentially) full control on the workload of our machine

Portable Batch System – Overview/1

- PBS from MRJ Technology Solutions (Veridian Corporation). This Pachage was developed by MRJ for the NAS Facility at NASA Ames Research Center; it's successor of the venerable NQS packages, which was also developed at NASA Aimes
- PBS is software system for managing system resources on workstations,SMP systems, MPPs, and vector supercomputers.
- Developed with the intent to conform with the POSIX Batch Environment Standard

Portable Batch System – Overview/2

- Available for everything that is UNIX-like
- Today PBS is the most widely used in Linux Cluster environments
- Two branches: OpenPBS (Open and Free - latest version is 2.3.16) and PBSpro, supported by Veridian, with some more features.

We will use the term “PBS” to indicate the Open version, but almost everything can be applied to the commercial one

How PBS Handles a Job

- User determines resource requirements for a job and writes a batch script
- User submits job to PBS with the *qsub* command
- PBS places the job into a queue based on its resource requests and runs the job when those resources become available
- The job runs until it either completes or exceeds one of its resource request limits
- PBS copies the job’s output into the directory from which the job was submitted and optionally notifies the user via email that the job has ended

Determining Job Requirements

- For single CPU jobs, PBS needs to know at least two resource requirements:
  - CPU time
  - Memory
- For multiprocessor parallel jobs, PBS also needs to know how many nodes/CPU are required
- Other things to consider:
  - Job name ?
  - Where to put standard output adn error output ?
  - Should the system email when the job completes ?

PBS job scripts /1

- A PBS job script is just a regular shell script with some comments (the ones starting with #PBS) which are meaningful to PBS. These comments are used to specify properties of the job.
- PBS job script always start in your home directory, $HOME. If you need to work in another directory, your job script will need to *cd* to there
- PBS jobs have all characteristics of a tipical UNIX session associated with them
  - ✓ A login procedure
  - ✓ stdin, stdout, stderr

PBS Job scripts / 2

Useful PBS options:

-l mem=N[KMG] (request N [kilo|mega|giga] bytes of memory)
-l cput=hh:mm:ss (max CPU time per job request)
-l walltime=hh:mm:ss (max wall clock time per job request)
-l nodes=N:ppn=M (request N nodes with M processors per node)
-I (run as an interactive job)
-N jobname (name the job *jobname*)
-S shell (use *shell* instead of your login shell to interprete the batch script)
-q queue (explicitely request a batch destination queue)
-o outfile (redirect standard output to *outfile*)
-e errfile (redirect standard error to *errfile*)
-j oe (combine stdout and stderr together)
-m e (mail the user when the job completes)

PBS script file example

Here is a simple batch job:
[cluster]$ cat xyz.job

```
#PBS -1 cput=40:00:00
#PBS -1 mem=36MB
#PBS -1 nodes=1:ppn=1
#PBS -N xyz
#PBS -j oe
#PBS -S /bin/ksh
cd $HOME/Beowu1f/xyz/
/usr/bin/time ./mybin > xyz.out
```

This job asks for one CPU on one node, 36MB of memory, and 40 hours Of CPU time. Its name is “xyz".

Submitting a job

Use the *qsub* command:

```
[cluster]$ qsub xyz.job
```

Options can also be specified on the command line:

qsub [-a date_time] [-A account_string] [-c interval] [-C directive_prefix] [-e path] [-h] [-I] [-j join] [-k keep] [-l resource_list] [-m mail_options] [-M user_list] [-N name] [-o path] [-p priority] [-q destination] [-r c] [-S path_list] [-u user_list] [-v variable_list] [-V] [-W additional_attributes] [-z] [script]

Monitoring a job

Use the *qstat* command. *qstat* let you monitor the job status (and not only!) in various ways.

qstat [-f] [-W site_specific] [job_identifier... | destination...]
qstat [-a|-i|-r] [-u user] [-n] [-s] [-G|-M] [-R] [job_id... | destination...]

Some interesting options are:

-f Specifies that a full status display be written to standard out.
-a "All" jobs are displayed in the alternative format.
-n In addition to the basic information, nodes allocated to a job are listed.

qstat -a example

[cluster]$ qstat -a

cluster.beowulf.cineca.it:

```
                                                                  Req'd  Req'd   Elap
Job ID          Username  Queue    Jobname    SessID NDS TSK Memory Time  S Time
--------------- --------- -------- ---------- ------ --- --- ------ ----- - -----
3492.cluster.be incpv402 dque     disper.tun  18218   1  --    --  05:00 R 02:27
3493.cluster.be incpv402 dque     disper.tun  16826   1  --    --  05:00 R 02:16
3494.cluster.be incpv402 dque     disper.tun     --   1  --    --  05:00 Q   --
3495.cluster.be incpv402 dque     disper.tun     --   1  --    --  05:00 Q   --
3500.cluster.be cmpfik50 dque     01-02-1.s      --   8  --    --  06:00 R   --
3502.cluster.be cmpfik50 dque     01-02-3.s   21443   8  --    --  06:00 R 01:49
3503.cluster.be cmpfik50 dque     01-02-4.s      --   8  --    --  06:00 Q   --
3552.cluster.be incpv402 dque     disper.tun     --   1  --    --  04:00 Q   --
3553.cluster.be incpv402 dque     disper.tun     --   1  --    --  04:00 Q   --
3566.cluster.be cmptskz0 dque     md28         5887  16  --    --  06:00 R 00:31
3574.cluster.be cmpnapa2 dque     glass_0010  20283  16  --    --  06:00 R 01:43
3575.cluster.be cmprmk80 dque     sottometti  17222  13  --    --  06:00 R 01:18
3576.cluster.be cmprmk80 dque     sotto2      20376  13  --    --  06:00 R 01:09
```


qstat –n example

```
[root@cluster server_priv]# qstat -n

cluster.beowulf.cineca.it:
                                                                  Req'd  Req'd   Elap
Job ID          Username Queue    Jobname    SessID NDS TSK Memory Time  S Time
--------------- -------- -------- ---------- ------ --- --- ------ ----- - -----
3749.cluster.be cmpnapa1 dque     ideargy_l3  24312  16  --    --  06:00 R 05:11
   node43/1+node42/1+node41/1+node40/1+node39/1+node38/1+node37/1+node30/1
   +node29/0+node28/0+node26/0+node25/0+node24/0+node23/0+node22/0+node20/0
3752.cluster.be cmprmk80 dque     sottometti  30449  13  --    --  06:00 R 04:45
   node62/0+node64/0+node63/0+node61/0+node60/0+node59/0+node58/0+node57/1
   +node56/1+node55/1+node54/1+node53/0+node52/0
3753.cluster.be cmprmk80 dque     sotto2      27966  13  --    --  06:00 R 03:20
   node64/1+node63/1+node61/1+node60/1+node59/1+node58/1+node57/0+node56/0
   +node55/0+node54/0+node53/1+node52/1+node51/0
```

How to see queue limits

qstat -Q [-f][-W site_specific] [destination...]
qstat -q [-G|-M] [destination...]

```
[amr0@cluster amr0]$ qstat -Q -f
Queue: dque
    queue_type = Execution
    total_jobs = 4
    state_count = Transit:0 Queued:0 Held:0 Waiting:0 Running:4 Exiting:0
    resources_max.walltime = 06:00:00
    resources_default.walltime = 02:00:00
    resources_assigned.nodect = 58
    enabled = True
    started = True

[amr0@cluster amr0]$ qstat -Q
Queue              Max Tot Ena Str Que Run Hld Wat Trn Ext Type
---------------- --- --- --- --- --- --- --- --- --- --- ----------
dque                 0   4 yes yes   0   4   0   0   0   0 Execution
[amr0@cluster amr0]$ qstat -q

server: cluster

Queue            Memory CPU Time Walltime Node Run Que Lm  State
---------------- ------ -------- -------- ---- --- --- --  -----
dque               --      --    06:00:00  --    4   0 --   E R
                                                --- ---
                                                  4   0
```


Killing a job

qdel *job_id*

- Let you remove a queued job or a running job. The “job_id” can be obtained from the *qstat* command. All users, apart from the PBS administrators, can delete only their jobs.

Example:

```
[cluster]$ qdel 3124
```

Interactive Batch Job support

PBS supports the option of an Interactive Batch Job for debugging purposes through PBS directives

General limitations:

- qsub command reads standard input and passes the data to the job, which is connected via a pseudo tty
- PBS only handles standard input, output and error
- Compute nodes are on a private network

Interactive Batch Setup

To run an interactive job a user would have a qsub script that only contains a preamble, for example:

```
#PBS -N nblock
#PES -j oe
#PBS -l nodes=1:ppn=1
#PBS -l cput=1:00:00
#PBS -l mem=256MB
```

Notice there are no shell script commands in the qsub script. The user would submit the script with the command:

```
[cluster]$ qsub -I script-name
```

The extra “I” flags denote interactive request.

Pitfalls to the interactive jobs

- Job requests are normally scheduled
- Users could get impatient and leave their terminal
- Users could forget to log out of the interactive batch shell
- Makes accounting challenging

SMP jobs

At least from PBS's point of view, the only difference between a uniprocessor and an SMP job is the resource request limit

```
-l nodes=1:ppn=N
```

with N>1 , contained in the SMP job script.

This tells PBS that the job will run N processes (or threads) concurrently on one node, so PBS allocates the required CPUs for you.

If you simply request a number of nodes (eg. –l nodes=1), PBS will assume that you want one processor per node.

Parallel Jobs

Both serial and SMP jobs run on only 1 node. However, most MPI programs should be run on more than 1 node. Here is an example of how to do that:

```
#PBS -N nblock
#PBS -j oe
#PBS -l nodes=4:ppn=4
#PBS -l cput=1:00:00
#PBS -l mem=256MB
cd ~/Beowulf/mpi-c
mpirun.ch_gm ./nblock
```


Parallel job spawning

PBS provides a means by which a parallel job can spawn, monitor and control tasks on remote nodes.
Unfortunately , in general, no vendor has made use of this capability
⇒ spawing the tasks of a parallel job is a parallel environment duty:

- PVM provides one means by which a parallel job spawns processes via the pvmd daemon
- MPI typically has a vendor dependent method, often using rsh or rexec

All of these means are outside of PBS's control !!!!

- PBS cannot control or monitor resource usage of the remote tasks, only the ones started by the job on Mother Superior.
- PBS can only make the list of allocated nodes available to the parallel job and hope that the user make use of the list and stay within the allocated nodes. The names of the allocated nodes are placed in a file in ${PBS_HOME}/aux , whose name is passed to the job in the environment variable **PBS_NODEFILE**.

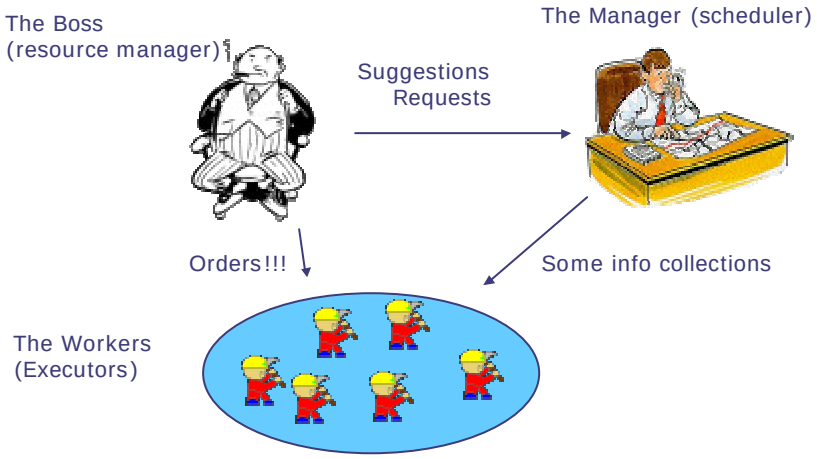
PBS structure /1

General components
- A resource manager (PBS server)
- A scheduler (PBS scheduler)
- Many “executors” (PBS moms)

Daemons communicate through TCP sockets

PBS provide an API to communicate with the server and another interface to interface the moms

PBS structure /2



PBS structure /3

PBS Server

- There is one server process
- It creates and receives batch jobs
- Modifies batch jobs
- Invokes the scheduler
- Instruct moms to execute jobs

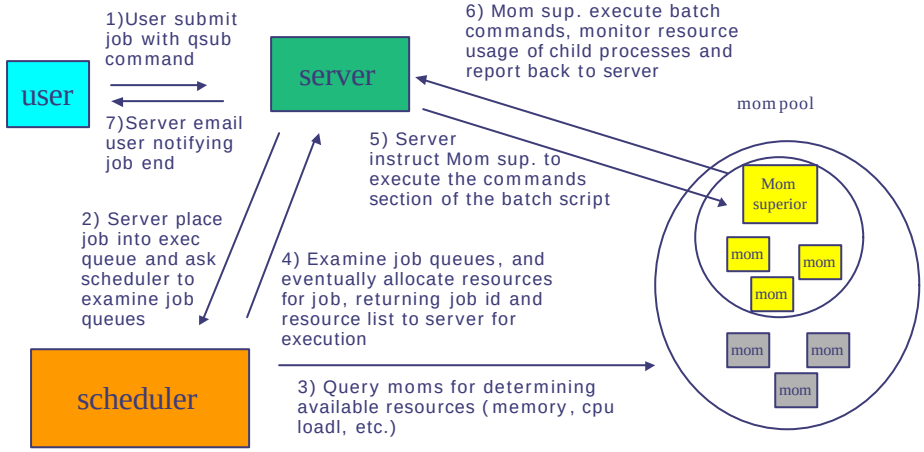
PBS Scheduler

- There is one scheduler process
- Contains the policy controlling which job is run, where and when it is run
- Communicates with the "moms" to learn about state of system
- Communicate with server to learn the availability of jobs

PBS MOM (Machine Oriented Miniserver)

- One process required for each compute node
- Places jobs into execution
- Takes instruction from the server
- Requires that each instance have its own local file system

A tipical job session



Obtaining PBS

OpenPBS is freely available from
http://www.openpbs.org

You just have to fill in a registration form and download the tarball file.

Rpm packages are available, but sources are better for configuration customization.

Building PBS

- Extract the files from tarball downloaded

```
# mkdir /usr/local/src/pbs
# cd /usr/local/src/pbs
# tar xvf $PATH_OF_DOWNLOAD/OpenPBS_2_3_12.tar.gz
```

- Move into the PBS directory and run the configure script with the appropriate option, and finally run make:

```
# cd OpenPBS_2_3_12
# ./configure --prefix=/usr/local/pbs --enable-docs --with-scp
  --set-server-home=/var/spool/pbs
# make
```

If using RedHat 7.1, you have a compilation error due to headers location mismatch. Please substitute in
<sorce-tree> /src/resmom/linux/mon_mach.c
the lines:

```
#include <linux/quota.h>
#include <sys/time.h>
```

with

```
#include <sys/quota.h>
#include <time.h>
```

Installing PBS /1

- As root, on the master node, do:
  ```
  # make install
  ```
- Executables will be installed under the directory tree:
  `/usr/local/pbs`
  and the configuration files into the server home directory $PBS_HOME:
  `/var/spool/pbs`
- The installation directory should be shared with all of the compute nodes, either by copying or (preferably) using a shared NFS filesystem.
- The `/usr/local/pbs/bin` directory should be added to users' PATH.

Installing PBS /2

- The *pbs_mom* running on each compute node needs its own copy of the *server_home* directory tree

=> copy it out to all of your compute nodes. The needed directories under the $PBS_HOME tree are essentially :

```
drwxr-xr-x    2 root     root         4096 Jan 27 05:51 aux
drwxr-xr-x    2 root     root         4096 Dec 28 23:26 checkpoing
drwxr-xr-x    2 root     root         4096 Jan 27 00:16 mom_logs
drwxr-x--x    3 root     root         4096 Dec 28 23:27 mom_priv
-rw-r--r--    1 root     root           30 Dec 28 23:26 pbs_environment
-rw-r--r--    1 root     root            8 Dec 28 23:26 server_name
drwxrwxrwt    2 root     root         4096 Jan 27 05:51 spool
drwxrwxrwt    2 root     root         4096 Dec 28 23:26 undelivered
```

Permission settings are important!!

- PBS as shipped assumes a consistent user name space within the set of systems which make up a PBS cluster.

Installing PBS /3

For completeness, add the following to your /etc/services file:

```
pbs             15001/tcp  # pbs server (pbs_server)
pbs_mom         15002/tcp  # mom to/from server
pbs_resmom      15003/tcp  # mom resource management requests
pbs_resmom      15003/udp  # mom resource management requests
pbs_sched       15004/tcp  # scheduler
```


PBS Server

- Configuring the server can be separated into two parts:
  - ✓ Configuring the server attributes
  - ✓ Configuring queues and their attfibutes
- Server is configured with the *qmgr* command while it is running

Usage: **qmgr [-c cormand] -n**

-c Execute a single command and exit qmgr

-n No commands are executed, syntax checking only is performed

- Commonly used commands:
  - set, unset, print, create, delete, quit
- Commands operate on three main entities
  - server set/change server parameters
  - node set/change properties of individual nodes
  - queue set/change properties of individual queues

Server attributes /1

Default queue

- Declares the default queue to which jobs are submitted if a queue is not specified
- usually clusters are structured so that all jobs go first though a routing queue and then to the specific destination queue
- This routing queue is usually the default queue

  set server default_queue = routing_queue

Access Control List (ACL)

- Hosts: a list of hosts from which jobs may be submitted

  set server acl_hosts = *.cineca.it

  set server acl_host_enable = True

  ✓ True=turn this feature on

  ✓ False=tum this feature off
- Users: a list of users who may submit jobs

  set server acl_user = wrkOO1@*,wrkOO2@*,wrkOO3@*

  set server acl_user_enable = True

Server attributes /2

Managers

- Defines which users at a specified host are granted batch system administration privilege

  ✓ set server managers = admin0l@*.cineca.it

  ✓ set server managers += pinky@*.cineca.it

  ✓ set server managers += brain@*.cineca.it

Node Packing

- Defines the order in which multiple cpu cluster nodes are allocated to jobs

  ✓ True: jobs are packed into the fewest possible nodes

  ✓ False: jobs are scattered across the most possible nodes

  set server node_pack = True

Query Other Nodes

- True: qstat allows you to see all jobs system
- False: qstat only allows you to see your jobs

  set server query_other_jobs = True

Node attributes

Node creation and attributes can be entered:

- using the *qmgr* program while the server is running
- editing a "nodes" file in `$PBS_HOME/server_priv/` before the server is started. The format of this file is the following:

node_name [property …] [np=N] [property …]

where "property" is a string, and N is the number of virtual processors which can be allocated on that node. Here is an example:

```
[root@cluster server_priv]# more nodes
node01 np=2 all rack01 compute cisco02 mgasma01 pos0101 myri gpfs
node02 np=2 all rack01 compute cisco02 mgasma01 pos0102 myri gpfs
node03 np=2 all rack01 compute cisco02 mgasma01 pos0103 myri gpfs
…
node33 np=2 all rack02 compute cisco01 mgasma05 pos0201 myri
node34 np=2 all rack02 compute cisco01 mgasma05 pos0202 myri
node35 np=2 all rack02 compute cisco01 mgasma05 pos0203 myri
```


Queue structure

PBS defines two types of queues

- Routing (route)
  - ✓ Used to move jobs to other queues
  - ✓ Jobs cannot execute in a routing queue
- Execution (execution)
  - ✓ A job must reside in an execution queue to be eligible to run
  - ✓ Job remains in this queue during execution

Usual queue configuration

- One routing queue that is the entry point for all jobs
- Routing queue dispatches jobs to execution queues defined by cpu time and number of processors requested

PBS queue attributes

Queue attributes are configured with the `qmgr` command while it is running

Usage:

```
[cluster@root]# qmgr
Qmgr: create queue queue_name
set queue queue_name attribute_name = value
```

see *man pbs_queue_attributes* for a complete list of queue attributes

Before queue attributes can be set, the queue must be created:

```
create queue batch
create queue short_16pe
create queue long_16pe
```


Reccomended PBS queue attributes/1

Max running

- Controls how many jobs in this queue can run simultaneously
- Customize this value based on hardware resources available

```
set queue short_16pe max_running = 4
```

Max user run

- Controls how many jobs an individual userid can execute simultaneously across the entire server
- Helps prevent a single user from monopolizing system resources

```
set queue short_16pe max_user_run = 2
```

Priority

- Sets the priority of a queue relative to othe queues
- Provides a method of giving smaller jobs quicker turnaround

```
set queue q5p128 Priority = 90
```

Reccomended PBS queue attributes/2

Maximum and Minimum resources

- Limits can be placed on various resource limits.
- This restricts which jobs may enter the queue based on the resources requested

Usage:

```
set queue short_16pe resources_max.resource = value
```

Look at *man pbs_resources_linux* to see all resource attributes for linux:

cput	maximum amount of CPU time used by all processes
nodes	number of nodes to be reserved
ppn	number of processors to be reserved on each node
pmem	maximum amount of physical memory used by any single processes
walltime	maximum amount of real time during which the job can be in the running state

Example of PBS execution queue

```
create queue short_16pe
set queue short_16pe queue_type = Execution
set queue short_16pe Priority = 90
set queue short_16pe max_running = 8
set queue short_16pe resources_max.cput = 10:00:00
set queue short_16pe resources_max.nodect = 4
set queue short_16pe resources_max.nodes = 4:ppn=4
set queue short_16pe resources_min.nodect = 2
set queue short_16pe resources_default.cput = 05:00:00
set queue short_16pe resources_default.mem = 1900mb
set queue short_16pe resources_default.nodect = 4
set queue short_16pe resources_default.nodes = 4:ppn=4
set queue short_16pe resources_default.vmem = 1900mb
set queue short_16pe max_user_run = 4
set queue short_16pe enabled = True
set queue short_16pe started = True
```

PBS scheduler

PBS implements the scheduler as a module, so that different sites can “plug in” the scheduler that meets their specific needs

In the following, we will talk about the default PBS FIFO scheduler coming with the PBS distribution (also called “C scheduler”)

Default PBS scheduler/1

Features:

- All jobs in a queue will be considered for execution before the next queue is examined
- All queues are sorted by priority
- Within each queue, jobs are sorted by requested CPU time (jobs can be sorted on multiple keys)
- Jobs which have been queued for more than 24 hours will be considered starving

Default PBS scheduler /2

Configuring the scheduler

- Configuration file read when scheduler is started
- $PBS_HOME/sched_priv/sched_config
- FIFO scheduler will require some customization initially, but should remain fairly static

Format of config file

- One line for each attribute
  name: value { prime | non-prime | all }
- Some attributes will require a prime option
- If nothing is placed after the value, the default of "all" will be assigned
- Lines starting with a "#" will be interpreted as comments
- When PBS is installed, an initial sched_config is created

Scheduler attributes

strict_fifo

- Controls whether jobs will be scheduled in strict FIFO order or not
- Type: boolean
  - ✓ True - jobs will be run in a strict first in first out order
  - ✓ False - jobs will be scheduled based on resource usage

Help_starving_jobs

- Once a queuedjob has waited a certain period of time, PBS will cease running job's until the "starving job" can be run
- Waiting period for starving job status is defined in starv_max
- Type: boolean
  - ✓ True - starving job support is enabled
  - ✓ False - starving job support is disabled

PBS Mom

- Configuring the execution server (Mom) is achieved with a configuration file, which is read in at startup
- Configuration file location
  Default: `$PBS_HOME/mom_priv/config`

You can specify a different file with the '-c' option when the *pbs_mom* daemon is started

Configuration file contains two types of information:
- Initialization values and directives
- Static resources

Initialization Values and directives

$cputmult
- Sets a factor used to adjust the cpu time used by a job in case it might run on systems with different cpu performance

$usecp
- Declares which directories can be accessed with a simple "cp" copy instead of a remote (rcp or scp) copy. Useful for NFS home directories.

```
$usecp cluster.beowulf.cineca.it:/u /u
```

$clienthost
- Declares from which hosts pbs_mom can accept connections

```
$clienthost  cluster.beowulf.cineca.it
```

$logevent
- Declares which log verbosity must be used. 511 is maximum

```
$logevent 0x1ff
```

Static Resources

Static resources are names and values that you assign to a given node that identify its special characteristics.

The resources can be requested in the batch script if a job needs some special resource.

```
physmem     2009644
myrinet     2
```

Given the above definitions, jobs that want up to 2 gig of memory and 2 myrinet card "could" be scheduled on this node

If a job asked for 3 myrinet interfaces it could not be scheduled on this node

```
#PBS -1 nodes=1:myrinet=2   could be scheduled on this node
#PBS -1 nodes=1:myrinet=3   could not be scheduled on this node
```


Prologue/Epilogue scripts

PBS provides the ability to run a site supplied script before and/or after each job runs. This provides the capability to perform initialization or cleanup of resources.

- Prologue script runs prior to each job from the node where runs the Mother Superior, while Epilogue script runs after the end of each job
- The script names and path are

  `$PBS_HOME/mom_priv/prologue`

  `$PBS_HOME/mom_priv/epilogue`
- The scripts must be owned by root
- The scripts must have permissions

```
root   read/write/execute
group & world      none
```

Sample Epilogue script

```
[node01@root]# cat /var/spool/pbs/mom_priv/epilogue
#!/bin/bash
PBS_HOME=/var/spool/pbs
echo "Begin PBS Epilogue $(date)"
echo "Job ID:           $1"
echo "Username: $2"
echo "Group:            $3"
echo "Job Name: $4"
echo "Session:  $5"
echo "Limits:           $6"
echo "Resources:        $7"
echo "Queue:            $8"
echo "Account:          $9"
echo -n "Nodes:         "
for i in $(sort $PBS_HOME/aux/$1 | uniq)
do
        echo -n "$i "
done
```


Ready to run

- Configure and start the Moms on compute nodes

```
# pbs_mom -r
```

- Configure nodes file and start the server. For the first time, use the “-t create” argument to initialize the server DB

```
# pbs_server –t create
```

- Start the scheduler

```
# pbs_sched
```

- Set server attributes and create queues
- Set a default queue for server

```
qmgr> set server default_queue = your_default_queue
```

- Enable scheduling

```
# qmgr –c “set server scheduling=true”
```

PBS logs

Daemon logs can be found in:

```
$PBS_HOME/server_logs
$PBS_HOME/sched_logs
$PBS_HOME/mom_logs
```

named with the YYYYMMDD naming convention.
Accounting logs are found with the same naming convention in
`$PBS_HOME/server_priv/accounting`

PBS accounting log example

```
[cluster@root]# cat $PBS_HOME/server_priv/accounting/20020128
…
01/28/2002 00:08:31;Q;3763.cluster.beowulf.cineca.it;queue=dque
…
01/28/2002 00:08:31;S;3763.cluster.beowulf.cineca.it;user=cmptsk31
group=infmk3 jobname=opt.sh queue=dque ctime=1012172911 qtime=1012172911
etime=1012172911 start=1012172911
exec_host=node54/1+node54/0+node53/1+node53/0+node52/1+node52/0+node51/1+node
51/0+node34/1+node34/0+node33/1+node33/0+node32/1+node32/0+node31/1+node31/0
Resource_List.neednodes=node54:ppn=2+node53:ppn=2+node52:ppn=2+node51:ppn=2+n
ode34:ppn=2+node33:ppn=2+node32:ppn=2+node31:ppn=2
Resource_List.nodect=8 Resource_List.nodes=8:ppn=2
Resource_List.walltime=06:00:00
…
01/28/2002 02:32:18;E;3763.cluster.beowulf.cineca.it;user=cmptsk31
group=infmk3 jobname=opt.sh queue=dque ctime=1012172911 qtime=1012172911
etime=1012172911 start=1012172911
exec_host=node54/1+node54/0+node53/1+node53/0+node52/1+node52/0+node51/1+node
51/0+node34/1+node34/0+node33/1+node33/0+node32/1+node32/0+node31/1+node31/0
Resource_List.neednodes=node54:ppn=2+node53:ppn=2+node52:ppn=2+node51:ppn=2+n
ode34:ppn=2+node33:ppn=2+node32:ppn=2+node31:ppn=2
Resource_List.nodect=8 Resource_List.nodes=8:ppn=2
Resource_List.walltime=06:00:00 session=26337 end=1012181538 Exit_status=0
resources_used.cput=25:42:42 resources_used.mem=1783504kb
resources_used.vmem=284180kb resources_used.walltime=02:23:42
```

Possible Improvements

- The system sometimes hangs when some pbs_mom have problems

=> Consider applying the Sandia patch

- Default scheduler is an old-fashion scheduler, and lacks some feature of the newest scheduler (backfill scheduler, reservation, etc)

=> MAUI scheduler

- Default mpich library does not interface deeply with PBS, generating:
  - Access problems
  - Accounting inaccuracy
  - More complex scripting for mpich host files preparation

=> mpiexec

The Sandia patch

Two basic problems. From the Sandia patch documentation:

1. “At every scheduling cycle, the server sends a list of MOMs to the scheduler. The scheduler tries to contact each MOM to get resource information so it can make an intelligent scheduling decision. If the MOM or the MOM's node is no longer talking, the scheduler hangs for three minutes (or whatever number of seconds it's "-a" argument specified) and then takes an alarm and exits. With the patch, it can still happen, but far less likely.”

2. “The second problem is that the server hangs if it tries to contact a MOM on a dead node. The solution implemented here is to use non-blocking sockets and timeout with an error”

Sandia patch installation

Download from:
http://www.cs.sandia.gov/cplant/doc/pbs/pbs.html

Patch is used for PBS at Cplant, but can be applied also to other sites.
Just do:

```
# cd <src_PBS_top_tree>
# patch –N –p1 –l < cplantFRpatch
```

After that, rebuild PBS and reinstall it

The MAUI scheduler

Developed by the MAUI supercomputing center as an alternative to the default Loadleveler scheduler in their IBM SP environment

Porting has been done to support also
- OpenPBS
- Wiki

Porting has been done essentially using the appropriate API provided by the batch system.

The MAUI scheduler

The MAUI scheduler has:

- Backfill scheduling algorithm: wall clock time (=estimated job duration) is a required attribute for each job
- Reservations for high priority jobs
- More control parameters on users
- Commands for querying the scheduler

The MAUI scheduler has NOT a very accurate documentation…..

Backfill algorithm /1

Backfill is a scheduling optimization which allows a scheduler to make better use of available resources by running jobs out of order.

Consider this example with a 10 CPUs machine:

Job1 (priority =20 walltime =10 nodes=6)
Job2 (priority =50 walltime =30 nodes=4)
Job3 (priority =40 walltime =20 nodes=4)
Job4 (priority =10 walltime =10 nodes=1)

1) When Maui schedules, it prioritizes the jobs in the queue according to a number of factors and then orders the jobs into a 'highest priority first' sorted list.
Sorted list:

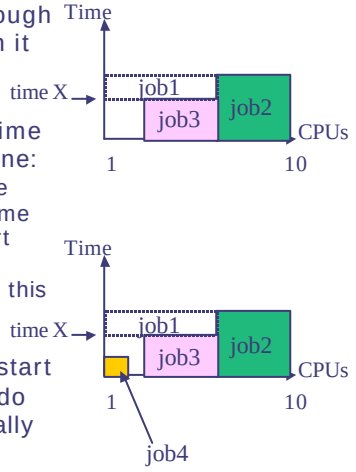
Job2 (priority =50 walltime =30 nodes=4)
Job3 (priority =40 walltime =20 nodes=4)
Job1 (priority =20 walltime =10 nodes=6)
Job4 (priority =10 walltime =10 nodes=1)

Backfill algorithm /2

2) It starts the jobs one by one stepping through the priority list until it reaches a job which it cannot start.

3) All jobs and reservations possess a start time and a wallclock limit, so Maui can determine:
- the completion time of all jobs in the queue
- the earliest the needed resources will become available for the highest priority job to start (time X).
- which jobs can be started without delaying this job (job4).

=> Enabling backfill allows the scheduler to start other, lower-priority jobs so long as they do not delay the highest priority job, essentially filling in holes in node space.

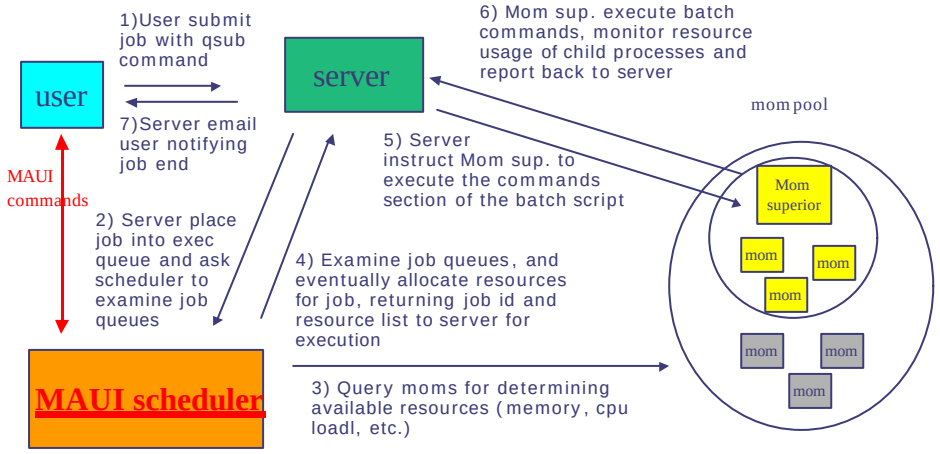


Backfill algorithm results

Backfill offers significant scheduler performance improvement:
- increased system utilization by around 20% and improved turnaround time by an even greater amount in a typical large system
- backfill tends to favor smaller and shorter running jobs more than larger and longer running ones: It is common to see over 90% of these small and short jobs backfilled.

⇒sites will see marked improvement in the level of service delivered to the small, short jobs and only moderate for the larger, long ones.

Change in job session scheme



MAUI user commands

showq
- Show job status and some job info

showbf, showbf -v
- Check for immediately available CPUs and nodes

checkjob *job_id*
- Check job status

canceljob *job_id*
- Cancel a job, sending essentially a “qdel” to the pbs_server

showstart *job_id*
- Show when job is scheduled to start

MAUI showq

```
[amr0@cluster mans]$ showq
ACTIVE JOBS--------------------
             JOBNAME USERNAME      STATE  PROC   REMAINING            STARTTIME

                3698 cmprmk80    Running    13     0:46:13  Sat Jan 26 11:23:10
                3705 incpv402    Running     1     1:34:44  Sat Jan 26 14:11:41
                3706 incpv402    Running     1     1:34:44  Sat Jan 26 14:11:41
                3703 cmprmkt0    Running    16     2:34:51  Sat Jan 26 13:11:48
                3699 cmprmk80    Running    13     3:11:27  Sat Jan 26 13:48:24
                3713 sisci001    Running    16     5:04:17  Sat Jan 26 15:41:14
                3714 cmptskz0    Running    16     5:50:33  Sat Jan 26 16:27:30
                3716 cmptskz0    Running    16     5:53:00  Sat Jan 26 16:29:57

     8 Active Jobs      92 of  126 Processors Active (73.02%)
                        61 of   63 Nodes Active      (96.83%)

IDLE JOBS----------------------
             JOBNAME USERNAME      STATE  PROC     WCLIMIT            QUEUETIME

0 Idle Jobs

NON-QUEUED JOBS----------------
             JOBNAME USERNAME      STATE  PROC     WCLIMIT            QUEUETIME

                3707 incpv402       Idle     1     4:00:00  Sat Jan 26 14:11:41

Total Jobs: 9   Active Jobs: 8   Idle Jobs: 0   Non-Queued Jobs: 1
```


MAUI showbf

```
[amr0@cluster mans]$ showbf -v
backfill window (user: 'amr0' group: 'cineca' partition: ALL) Sat Jan 26 16:37:56

 34 procs available with no timelimit


node node01 is available with no timelimit
node node02 is unavailable because it is in state 'Busy'
node node03 is unavailable because it is in state 'Busy'
node node04 is unavailable because it is in state 'Busy'
node node05 is unavailable because it is in state 'Busy'
node node06 is unavailable because it is in state 'Busy'
node node07 is unavailable because it is in state 'Busy'
node node08 is unavailable because it is in state 'Busy'
node node09 is unavailable because it is in state 'Busy'
node node10 is unavailable because it is in state 'Busy'
node node11 is unavailable because it is in state 'Busy'
node node12 is unavailable because it is in state 'Busy'
node node14 is unavailable because it is in state 'Busy'
node node15 is unavailable because it is in state 'Busy'
node node16 is unavailable because it is in state 'Busy'
node node17 is unavailable because it is in state 'Busy'
node node19 is unavailable because it is in state 'Busy'
node node20 is available with no timelimit
…
```

MAUI checkjob

```
[sisci001]$ checkjob 3713

State: Running  (User: sisci001  Group: sissa  Account: [NONE])
WallTime: 2:36:54  (Limit: 6:00:00)

QueueTime: Sat Jan 26 15:41:14
StartTime: Sat Jan 26 15:41:14

Total Tasks: 16

Req[0]  TaskCount: 16  Partition: DEFAULT
Network: [NONE]  Memory >= 0  Disk >= 0  Features: [NONE]
Opsys: [NONE]  Arch: [NONE]  Class: [dque 1]

Allocated Nodes:
[node19:2][node17:2][node16:2][node15:2]
[node14:2][node12:2][node11:2][node10:2]

IWD: [NONE]  Executable:  [NONE]
QOS: DEFAULT  Bypass: 0  StartCount: 1
Partition Mask: [ALL]
Flags:       RESTARTABLE

Reservation '3713' (-0:57:42 -> 5:02:18  Duration: 6:00:00)
PE:  16.00  StartPriority:  231
```


MAUI showstart

```
[amr0@cluster amr0]$ date;showstart 3827
Tue Jan 29 16:11:55 CET 2002
job 3827 requires 16 procs for 6:00:00
Earliest start is in          0:06:12 on Tue Jan 29 16:18:04
Earliest Completion is in     6:06:12 on Tue Jan 29 22:18:04
Best Partition: DEFAULT
```

MAUI administration commands

checknode *node_name*

- Check node status

diagnose, diagnose -p

- Show diagnostic info regarding various aspects, in particular jobs and job priorities

showstats

- Show MAUI statistics

showgrid *stat_type*

- Show a text-based table of MAUI statistics relative to *stat_type*

MAUI checknode

```
# checknode node01

State:         Busy  Opsys:         DEFAULT  Arch:       linux
Configured Resources: Procs: 2  Mem: 896  Swap: 2028  Disk: 1
Utilized   Resources: Procs: 2  Swap: 498
Dedicated  Resources: Procs: 2
Speed:      1.00  Load:       2.000
Frame:Slot: 1:1  Partition:  DEFAULT
Network:    [DEFAULT]
Features:
   [all][rack01][compute][cisco02][mgasma01][pos0101][myri][gpfs]
Classes:    [dque 2:2]

node has been in current state for 0:00:00

Reservations:
Job '3833'(x2)  -4:24:00 -> 1:36:00 (6:00:00)

Total Time: 16:03:30:36  Up: 16:02:38:20 (99.78%)  Busy: 4:04:23:02
(25.90%)

job '3833' running on node for 4:24:00
```

Maui diagnose

```
[root@cluster acorso01]# diagnose -p
diagnosing job priority information (partition: ALL)

Job                    PRIORITY*    Cred( User)  Serv(QTime:XFctr)
             Weights   --------        1(    1)     1(    2:  100)

3827                        835    -0.0(  0.0) 100.0(645.2:189.6)
3851                        317    -0.0(  0.0) 100.0(190.2:126.4)
3856                        288    -0.0(  0.0) 100.0(165.4:123.0)
3860                        228    -0.0(  0.0) 100.0(112.5:115.6)
3866                        164    -0.0(  0.0) 100.0( 55.8:107.7)
3867                        136    -0.0(  0.0) 100.0( 32.0:104.4)

Percent Contribution    --------     0.0(  0.0) 100.0( 61.0: 39.0)

* indicates system prio set on job
```


MAUI showstats

```
[root@cluster acorso01]# showstats
Maui running for       0:01:44  stats initialized on Fri Jan
   11 16:01:36

Eligible/Idle Jobs:                     6/25        (24.000%)
Active Jobs:                            7
Successful/Completed Jobs:          1150/1150       (100.000%)
Avg/Max QTime (Hours):              1.80/34.21
Avg/Max XFactor:                    0.00/32.76

Dedicated/Total ProcHours:      39825.36/48814.09   (81.586%)

Current Active/Total Procs:          112/118        (94.915%)

Avg WallClock Accuracy:            51.633%
Avg Job Proc Efficiency:         1597.770%
Est/Avg Backlog (Hours):             12.87/13.02
```

MAUI showgrid

```
# showgrid JOBCOUNT

Job Count (jobs)

[ PROCS ][ 0:15:00 ][ 0:30:00 ][ 1:00:00 ][ 2:00:00 ][ 4:00:00 ][ 8:00:00 ][ TOTAL ]
[    1  ][      90 ][      14 ][       1 ][      74 ][     100 ][---------][   279 ]
[    2  ][      70 ][      66 ][       6 ][      15 ][      68 ][---------][   225 ]
[    4  ][      10 ][      36 ][       2 ][      18 ][       9 ][---------][    75 ]
[    8  ][      12 ][      26 ][       9 ][       9 ][      65 ][---------][   121 ]
[   16  ][       4 ][      13 ][       3 ][       7 ][     321 ][---------][   348 ]
[   32  ][---------][       1 ][---------][       5 ][      49 ][---------][    55 ]
[   64  ][---------][---------][---------][---------][---------][---------][    19 ]
[  128  ][---------][---------][---------][---------][---------][---------][    28 ]
[ TOTAL ][     186 ][     158 ][      23 ][     157 ][     626 ][---------]
Total Jobs:                   1150
```


MAUI installation /1

Download from http://www.supercluster.org

From the “MAUI-PBS Integration Guide”:
Maui drives PBS via the PBS scheduling API. To enable Maui scheduling, the following steps must be taken:

1) Install PBS
   - keeping track of the PBS target directory, `$PBSTARGDIR`

2) Install MAUI
   - Become “root” (this is not necessary indeed but following configurations will be easier)
   - Extract MAUI sources from the distribution tarball file.
   - **cd** into the maui-<X> directory
   - run **./configure**, specifying the PBS target directory when queried (Maui requires a few PBS libraries and include files to enable the PBS scheduling interface)

MAUI installation /2

- If you have a non-standard PBS installation, You may need to modify **src/Makefile** and change **PBSIP** and **PBSLP** values and references as necessary for your local site configuration.

3) Configure PBS

- make $MAUIADMIN a PBS admin. Maui communicates with both pbs_server and pbs_mom daemons. The $MAUIADMIN should be authorized to talk to both PBS daemons.If you installed MAUI as root, no configuration changes are necessary.
- Activate PBS scheduling with the qmgr command
  set server scheduling=True

Seggestions:

- set PBS default queue
  qmgr –c 'set system default_queue = <QUEUENAME>'

MAUI configuration /1

Main config file is maui.cfg
The following parameters terminate the installation:

- Specify PBS as the resource manager. This should be taken care of by '**configure**', but if not, the following parameter must be specified in the **maui.cfg** file:
  RMTYPE[0] PBS
- If you have a non-standard PBS configuration, you may need to specify additional Maui parameters to point Maui to the right location:
  RMHOST[0] $PBSSERVERHOST
  RMPORT[0] $PBSSERVERPORT

You can check your installation starting maui and giving a 'showq' from the command line.

MAUI configuration /2

- A template maui.cfg comes with the distribution
- Default server port is 42559 , but can be changed with
  SERVERPORT <port_number>
- Many many parameters are available, and some are becoming rapidly obsolete with new releases
- Some parameters must still be set by PBS!!
- The only documentation repository is in www.supercluster.org, and is rapidly changing
- A mailing list is available for users

Mpiexec
the parallel job launcher for PBS

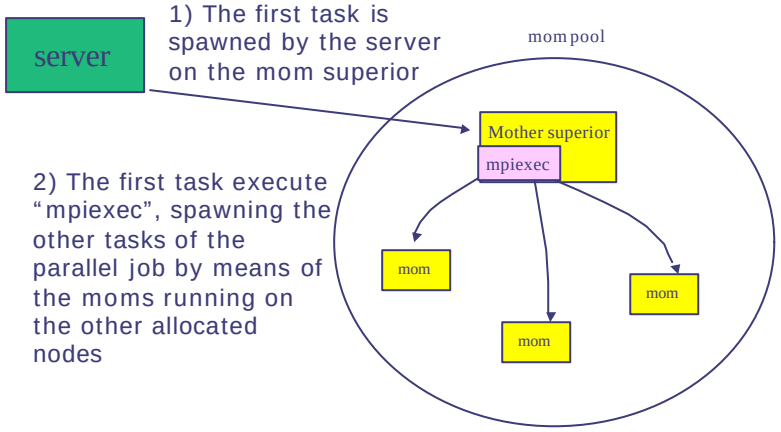
- Mpiexec is a replacement program for the script mpirun, which is part of the mpich package. It is used to initialize a parallel job from within a PBS batch or interactive environment.

- A few various implementations of message passing libraries are currently supported:
  - ✓MPICH/P4 - basic implementation of MPI over ethernet devices
  - ✓MPICH/GM - the above using the GM message-passing software on Myrinet

Mpiexec features

- Mpiexec uses the task manager library of PBS to spawn copies of the executable on (up to) all the nodes in a PBS allocation:
  - ✓ much faster than invoking a separate rsh/ssh once for each process
  - ✓ access problem is completely solved
  - ✓ resources used by the spawned processes are accounted correctly with mpiexec, and reported in the PBS logs
- Mpiexec handles creation of the node list file, if required by the message passing library, and the shared-memory file for use on SMP nodes.
- It also redirects standard input and output to the shell from which it was invoked, bypassing the PBS output and error files if you choose

How mpiexec works



Mpiexec usage

Instead of mpirun.ch_gm, simply use “mpiexec” with the following options:

mpiexec [OPTION]... executable [args]...

The basic options are:

-n numproc
Use only the specified number of processes. Default is to use all which were provided in the pbs environment.

-no-shmem (MPICH/GM and SMP only)
Instruct GM not to use shared memory communications between processes on the same SMP node. Using shared memory (the default) has generally higher throughput and always lower latency, but might result in heavy cache misses.

Mpiexec usage examples

```
# mpiexec a.out
```

Run the executable *a.out* as a parallel mpi code on each process allocated by pbs.

```
# mpiexec -n 2 a.out -b 4
```

Run the code with arguments -b 4 on only two processors.

Mpiexec installation / 1

Download tarball from:
http://www.osc.edu/~pw/mpiexec/
Tarball contains
- A patch for PBS
- Mpiexec sources

Unpack the tarball, generating a
`<src_mpiexec_top_tree>`
First of all, it's necessary to patch PBS:

```
# cd <src_PBS_top_tree>
```

For PBS releases >= 2.3.11:

```
# patch -p1 -sNE < <src_mpiexec_top_tree>/pbs-
2.3.11-mpiexec.diff
```

Now you can rebuild PBS

Mpiexec installation / 2

For what concerns mpiexec:

```
# cd <src_mpiexec_top_tree>
```

Then the usual "configure" script:

```
# ./configure –prefix=/usr/local/bin --
with-pbs=/usr/local/pbs --with-
pbssrc=/usr/local/src/OpenPBS_2_3_12  -
-with-smp-size=2  --with-mpich-gm --
with-myri-cards=1
# make
# make install
```